Recent Simulation Tune-Ups

Ben jones, MIT

LArG4 Speed Issues

- A current priority for the simulation is to get the memory usage and time requirements down.
- LArG4 is the slowest part of the simulation chain for most events. Lots of work has been done to reduce its memory consumption and time requirements.
- I made a few changes to increase the efficiency of geometry handling and improve simulation performance

Int NearestChannel(double* xyz)

Function :

Given a charge deposit somewhere in space, find the closest wire (called 10,000s of times per job)

Previous implementation Summary :

Scan through all wire objects loaded in geometry and calculate distance from each. Return the shortest

New implementation Summary :

Calculate the projection in the pitch direction, and given a knowledge of the wire pitch and first wire position, get the wire number

A couple of implementation details

- All variables from geometry are looked up only on the first call. As many calculations as possible are performed once only.
- As well as accepting the position of the deposit, the function also now accepts the plane and tpc ID and finds the nearest channel in that plane
- The output is range checked for invalid wire numbers, which generate an exception.
- Full wire number calculation can be performed in one line of code using all precalculated quantities

ChannelToWire and PlaneWireToChannel

Function :

Convert {wire ID, plane ID, TPC ID} to / from channel ID

Previous implementation Summary :

Scan through all wire objects loaded in geometry and find the one which matches one labelling specfication. Return the other

New implementation Summary :

Assume wires are labeled hierarchically (counting up from 0, tpc1 { plane1, plane 2…} tp2{ plane 1, plane 2…} etc. Calculate coordinates using statically stored boundaries between each plane

New geometry methods

- For all described so far, we assume uniformly spaced tpc wires in each plane with heirachical numbering.
- Might want to relax this in some weird future experiment so I also left the old brute force methods in place.
- The new methods are named ***Geometry::NearestChannelFast, Geometry::PlaneWireToChannelFast, Geometry::ChannelToWireFast***

LArVoxelReadout

Function :

Perform charge drift simulation for each charge deposited in LArG4

Previous implementation Summary :

Calculate charge diffusion, losses etc for each electron cluster and deposit on nearest wires in each plane

New implementation Summary :

General streamlining (calculating everything in the outermost for loop possible, statically store repeated results, etc) and modify to work with new geometry lookups

sim::SimChannel

Function :

Keep record of charge clusters drifted to each wire

Previous implementation Summary :

Store a vector of IDE objects recording the charges accepted for each tick and for each channel

New implementation Summary :

Mostly the same, but all charge passed to wires at end of job to reduce time spent looking up the right channel / time bin per cluster, and general code streamlining

GDML Geometry

- Within LArSoft, the geometry object given to LArG4 is distinct from that used to determine wire geometry for drift simulation – one comes from a root file and one from a gdml file (though origin is shared)
- It is not necessary for LArG4 to know about the wires at all – put in switch to allow us to feed LArG4 a wire-free geometry which greatly reduces overhead
- On Brians request, removed option to have both wired and wire free gdml available in the job. I am not sure which way I feel about this yet.
- Need to add automatic wireless-geometry building to geometry scripts (is Adam in this meeting?)

Validation

- Ran GeometryTest.cxx with new wire lookups and new geometry specification – works exactly as before and passes all validations
- Ran Brians MC cheater and found a 100% purity and efficiency of hits produced, given the MC charge deposits
- Cross checked all histograms from FFTHitAna between old code and new code, and all look identical (didn't include them here because I don't know what they all mean)
- If I have missed something and your simulation is weird, let me know.

MCCheater Output

ArgoNeuT Geometry

- Saima reported a warning message thrown by NearestChannelFast
- Apparently a known bug in ArgoNeuT geometry due to a gap in the wireplane – old method ignored the problem and double counted charge on outer wires.
- New method sees absence of wire coverage in TPC volume and declares an exception
- Is this warning message useful? Should drift to no wire region be allowed or not?

Performance enhancement:

- Running prod_single.fcl for uboone (note: not prod_single_uboone.fcl)

Old Code:

```
TimeReport> Time report
complete in 1422.83 seconds
 Time Summary:
 Min: 113.322
 Max: 296.957
 Avg: 142.283
```

New Code:

```
TimeReport> Time report
complete in 247.019 seconds
 Time Summary:
 Min: 18.3711
 Max: 30.4003
 Avg: 24.7019
```

Rest of modules take a fairly constant 8 seconds

Old LArG4 speed : 134 s / event
(approx 105s/event after first)
New LArG4 speed: 16 s / event
(approx 11s/event after first)